REDUCEによる非線型差分方程式の保存密度の計算 (数式処理における理論と応用の研究)

AUTHOR(S):

高, 敏; 加藤, 泰幸; 伊藤, 雅明

数理解析研究所講究録
1199巻 2001年 167-172

REDUCEによる非線型差分方程式の保存密度の計算

広島大学大学院工学研究科　高　敏(Min GAO) *
広島大学大学院工学研究科　加藤　泰幸(Yasuyuki KATO) †
広島大学工学部　伊藤　雅明(Masaaki ITO) ‡

1 はじめに

時間・空間の両変数に対して離散化された非線形差分方程式の保存密度の系列を求めることは、その方程式自身の可積分性を知る上で重要であるばかりでなく、現実の現象を記述する偏微分方程式の差分スキームの安定性を議論する上でも欠かせないものである。本研究においては、多項式型偏差分方程式の保存密度を解析的に求めるアルゴリズムを示し、それを数式処理システム（REDUCE）上にインプリメントした。開発したプログラムは、連立偏差分方程式に対してランクを自動的に決め、必要なランクの保存密度を具体的な形で求めるものである。このプログラムはランクが一様でない方程式にも適用することができる。

2 保存密度

従属変数 $\boldsymbol{u}(x,t)=(u^{(1)},u^{(2)},\ldots,u^{(N)})^t$ は、ある (x,t) を基準点として、x に対しては h の、t に対しては δ の刻み幅の整数 n,m 倍だけずれた点 $(x+nh,t+m\delta)$ における値をとるものとする。以後 $\boldsymbol{u}(x+nh,t+m\delta)$ を $\boldsymbol{u}_{n,m}$ と表わすことにする。

このように時間・空間の両変数とも離散化された $\boldsymbol{u}_{n,m}$ が方程式

$$\Delta_t \boldsymbol{u}_{n,m}=\boldsymbol{F}(\boldsymbol{u}_{n,m},\boldsymbol{u}_{n\pm1,m},\boldsymbol{u}_{n\pm2,m},\ldots), \tag{1}$$

*gaomin@amath.hiroshima-u.ac.jp
†ykato@amath.hiroshima-u.ac.jp
‡ito@amath.hiroshima-u.ac.jp

を満たしているとする。ここで、Δ_t は

$$\Delta_t f_{n,m} = \delta^{-1}(f_{n,m+1} - f_{n,m}) \tag{2}$$

で定義されている時間差分演算子である。また $\boldsymbol{F}$ は $\boldsymbol{u_{n,m}}$ とその x に関するシフト（ $\boldsymbol{u}_{n\pm1,m}$, $\boldsymbol{u}_{n\pm2,m}, \ldots$）の多項式であると仮定する。このとき、

$$\Delta_t \rho_n(\boldsymbol{u}_{n,m}, \boldsymbol{u}_{n\pm1,m}, \ldots) + \Delta_n J_n(\boldsymbol{u}_{n,m}, \boldsymbol{u}_{n\pm1,m}, \ldots) = 0 \tag{3}$$

は離散系における保存則を表わしている。ここで Δ_n は

$$\Delta_n f_{n,m} = h^{-1}(f_{n+1,m} - f_{n,m}) \tag{4}$$

で定義される空間差分演算子である。ρ_n は保存密度を、J_n は流束を表わしている。また、(4) 式を用いると、保存則 (3) は

$$\Delta_t \rho_n = h^{-1}(J_n - J_{n+1})$$

と表わすこともできる。

$\sum_n \rho_n$ が保存量となることを確かめるには、これの時間差分を計算してみればよく、

$$\begin{aligned} \Delta_t \sum_n \rho_n &= \sum_n \Delta_t \rho_n \\ &= h^{-1} \sum_n (J_n - J_{n+1}) \end{aligned} \tag{5}$$

となる。ここで、J_n が周期的境界条件 $J_{n+k} = J_n$ を満たすか、$|\, n \,| \to \infty$ において 0 となれば、$\sum_n (J_n - J_{n+1}) = 0$、即ち $\Delta_t \sum_n \rho_n = 0$ となり、$\sum_n \rho_n$ は時間的に不変な量（保存量）であることがわかる。以後簡単のために、誤解が生じない限り $\boldsymbol{u}_{n,m}$ を $\boldsymbol{u}_n$ と表し、必要に応じて時刻に対応するサフィックスを付けることにする。

次に、保存密度を求めるアルゴリズムで用いる同値関係について述べる。２つの単項式 M と N に対して、$N = S^d M = M\,|_{n \to n+d}$ $(d \in \mathbb{Z})$ であれば、M と N は同値であるといい、$M \equiv N$ と表わす。このとき、$M - N = \Delta_n P_n$ と表せる。ここで、S は $shift$ 演算子で、P_n は多項式である。また、単項式の変数を辞書式順序に並べたとき、その主変数のラベルの最小値を n にしたものを同値類の代表形とする。例えば、同値類 $\{\ldots, u_{n-1}v_{n-2}, u_n v_{n-1}, u_{n+1}v_n, \ldots\}$ を考えると、その代表形は $u_n v_{n-1}$ である ($u_{n+1}v_n$ ではない)。

3 保存密度探索のアルゴリズム

ここで用いるアルゴリズムは任意個数の未知変数に対して有効であるが、例として、次のような2変数差分方程式

$$\begin{cases} \Delta_t u_n &= v_n, \\ \Delta_t v_n &= u_{n-1}v_{n+1}, \end{cases} \tag{6}$$

を考える．アルゴリズムは本質的には一様なランクを持つ方程式の保存密度を求めるもので、次の3つのステップからなっている。基本的には偏微分方程式の保存量を求めるアルゴリズム [1,2,3] と同じであるが、差分多項式の生成には、Goktas 達 [4] のアルゴリズムを用いる。

ステップ1：与えられた方程式のランク（$rank$）及び変数の重みを決める

そこでまず、離散系におけるランクを定義しておく。
次のような単項式

$$f = \prod_{j=1}^{N}\prod_{i=1}^{k}(u^{(j)}_{n+l_i,m})^{d_{i,j}} \qquad (l_i, d_{i,j} \in \mathbb{Z})$$

に対してランクは

$$\begin{aligned} R(f) &= \sum_{j=1}^{N}\sum_{i=1}^{k} d_{i,j} w(u^{(j)}_{n+l_i,m}) \\ &= \sum_{j=1}^{N}\sum_{i=1}^{k} d_{i,j} w(u^{(j)}_{n,m}) \end{aligned} \tag{7}$$

と定義する。ここで、$w(u^{(j)}_{n,m})$ は $u^{(j)}_{n,m}$ の重みである。

多項式中のすべての項（単項式）のランクが同じであるとき、その多項式は一様なランクであると言う。方程式のランクが一様となるように各変数の重みを選ぶには、パラメータ λ を導入し、各変数を λ^k でスケール変換を行い、方程式が不変になるように k を決めればよい．方程式 (6) の場合、次のスケール変換

$$(\delta, u_n, v_n) \to (\lambda^{-1}\delta, \lambda u_n, \lambda^2 v_n)$$

を行うと方程式は不変となり、$w(u_n) = 1$, $w(v_n) = 2$, $w(\delta^{-1}) = w(\Delta_t) = 1$ が得られる。

ステップ2：ランク r の多項式（未定係数 c_i を含む）を生成する

• u_n のみを含むランク r 以下の単項式の集合 $G = \{\ldots, \Pi_{j=1}^{l}(u_n^{(j)})^{d_j}, \ldots\}$ $(\sum_{j=1}^{l} d_j w(u_n^{(j)}) \leq r)$ を生成する。

• 集合 G の各要素に適当な回数の時間差分を行い、ランクが r となる単項式の集合 H を作る。

• 集合 H の中から同値な項を取り除き、代表形で置き換えた集合 K を求め、K の各要素に未定係数 c_i を付けて ρ_n の候補を作る。

方程式 (3) に対してランク 3 の場合について上記の操作を行うと、G, H, K, ρ_n はそれぞれ次のようになる。

$$\begin{aligned} G &= \{u_n^3, u_n^2, u_n v_n, u_n, v_n\}, \\ H &= \{u_n^3, u_n v_n, u_{n-1} v_{n+1}, \delta v_n^2\}, \\ K &= \{u_n^3, u_n v_{n+2}, u_n v_n, \delta v_n^2\}, \\ \rho_n^{(3)} &= c_1 u_n^3 + c_2 u_n v_{n+2} + c_3 u_n v_n + c_4 \delta v_n^2. \end{aligned}$$

ステップ 3: $\Delta_t \rho_n + \Delta_n J_n = 0$ となるように、未定係数 c_i を決める

ステップ 2 で生成されたランク r の多項式 ρ_n に Δ_t を演算する. ランク 3 の場合、次式が得られる。

$$\begin{aligned} \Delta_t \rho_n^{(3)} &= 3c_1 u_n^2 v_n + 3c_1 \delta u_n v_n^2 + c_1 \delta^2 v_n^3 + c_2 u_n u_{n+1} v_{n+3} \\ &\quad + c_2 \delta u_n v_{n-1} v_{n+2} + c_2 v_n v_{n+2} + c_3 u_n u_{n+1} v_{n+2} \\ &\quad + c_3 v_n^2 + (c_3 + 2c_4) \delta u_n v_{n+1} v_{n+2} + c_4 \delta^2 u_n^2 v_{n+2}^2 \\ &\quad + [J_n - J_{n+1}] \end{aligned}$$

ここで、

$$\begin{aligned} J_n &= -c_2 \delta u_n v_{n-1} v_{n+2} + c_3 u_{n-1} u_n v_{n+1} + (c_3 + 2c_4) u_{n-1} v_n v_{n+1} \\ &\quad + c_4 \delta^2 u_{n-1}^2 v_{n+1}^2 \end{aligned}$$

である。ρ_n が保存密度であるためには、保存則 $\Delta_t \rho_n + \Delta_n J_n = 0$、または $\Delta_t \rho_n = h^{-1}(J_n - J_{n+1})$ を満たさなければならないので、等式右辺の $[J_n - J_{n+1}]$ 以外の項は 0 にならなければならない。つまり、それらの係数がすべて 0 にならなければならない。よって、次のような条件式が得られる。

$$S = \{3c_1 = 0, 3\delta c1 = 0, \delta^2 c_1 = 0, c_2 = 0, \delta c2 = 0, c_3 = 0, \delta(c_3 + 2c_4) = 0, \delta^2 c_4 = 0\}$$

この例の場合、これらの条件を満たす解は $c_1 = c_2 = c_3 = c_4 = 0$ となり、この方程式に対してはランク 3 の保存密度は存在しないことがわかる。

方程式のランクが一様でないときは、適当な重みをもった補助的なパラメータを導入し、以上と同様の操作を行う。

4 パッケージの構成

このパッケージにおいて代数モードから利用できるコマンドはWEIGとADCDの2つがあり、WEIGはステップ1に対応し、与えられた方程式のランクおよび変数の重みを決める。ADCDはステップ2、3に対応し、与えられた方程式に対してランクrの保存密度を求める。アルゴリズム中で使用している変数u_{n+l}はプログラム中では$u(l)$で表す。次のような方程式

$$\begin{cases} \Delta_t u1_{n,m} &= u1_{n,m}u2_{n+1,m}u3_{n+2,m} - u2_{n,m}u3_{n+1,m}, \\ \Delta_t u2_{n,m} &= u2_{n,m}u3_{n+1,m} - u3_{n,m}u1_{n+1,m}u2_{n+2,m}, \\ \Delta_t u3_{n,m} &= u3_{n,m}u1_{n+1,m}u2_{n+2,m} - u1_{n,m}u2_{n+1,m}u3_{n+2,m}, \end{cases} \tag{8}$$

に対しては、以下のようなコマンド実行すればよい。

```
dtu1:=u1(0)*u2(1)*u3(2)-u2(0)*u3(1)$
dtu2:=u2(0)*u3(1)-u3(0)*u1(1)*u2(2)$
dtu3:=u3(0)*u1(1)*u2(2)-u1(0)*u2(1)*u3(2)$
weig(dtu1,dtu2,dtu3)$
adcd(dtu1,dtu2,dtu3,3)$
```

結果は以下のとおりである。

```
******  original  ******

dtu1 = u1(0)*u2(1)*u3(2) - u2(0)*u3(1)
dtu2 =  - u1(1)*u2(2)*u3(0) + u2(0)*u3(1)
dtu3 = u1(1)*u2(2)*u3(0) - u1(0)*u2(1)*u3(2)

******  with parameter  ******

dtu1 = u1(0)*u2(1)*u3(2) - u2(0)*u3(1)*p1
dtu2 =  - u1(1)*u2(2)*u3(0) + u2(0)*u3(1)*p2
dtu3 = u1(1)*u2(2)*u3(0) - u1(0)*u2(1)*u3(2)

******  weight and rank  ******
```

```
weight of dt = 2
weight of u1 = 1      rank of dtu1 = 3
weight of u2 = 1      rank of dtu2 = 3
weight of u3 = 1      rank of dtu3 = 3
weight of p1 = 1
weight of p2 = 1

++++++ conserved density of rank(3) ++++++

                3          2                2                          2
cd!#(1) = u1(0)  + 3*u1(0) *u2(0) + 3*u1(0) *u3(0) + 3*u1(0)*u2(0)
                                                  2         3
          + 6*u1(0)*u2(0)*u3(0) + 3*u1(0)*u3(0)  + u2(0)
                    2                       2         3
          + 3*u2(0) *u3(0)+ 3*u2(0)*u3(0)  + u3(0)
```

参 考 文 献

[1] M. Ito and F. Kako, Comput. Phys. Commun. **38** (1985) 415.

[2] M. Ito, Comput. Phys. Commun. **79** (1994) 547.

[3] U. Goktas and W. Hereman, J. Symb. Comp. **24** (1997) 591

[4] U. Goktas, W. Hereman and G. Erdmann, Phys. Lett. A **236** (1997) 30.